# TEST PLAN TEMPLATE

*The format and content of a software test plan vary depending on the processes, standards, and test management tools being implemented. Nevertheless, the following format, which is based on IEEE standard for software test documentation, provides a summary of what a test plan can/should contain.*

## Test Plan Identifier:

Test Plan ID: HOA\_MANAGEMENT\_TP\_01

# Introduction:

## Overview

## This test plan defines the strategy, scope, approach, resources, and timeline for testing the HOA Management System. The primary goal is to ensure the system meets functional, usability, performance, and security requirements as outlined in the Project Requirements document. Built with React, Node.js, and MySQL, testing will emphasize these core components.

## Goals and Objectives

Testing will confirm a secure, efficient, and user-friendly system that supports HOA homeowner data management, seamless communication, role-based access, and HOA-specific functions like violations tracking, assessments, announcements, surveys, and leadership transitions.

* Verify compliance with all functional requirements
* Ensure data integrity and robust security throughout operations
* Validate a responsive and accessible user interface
* Confirm integration between frontend and backend components

## Constraints

The testing environment will replicate the production environment, with the exception of the application being on a live server. Time constraints limit testing to those components, the database that has been created and other packages that are utilized in the application. Third party software, applications and packages functionality will not be tested, only their functionality within the HOA Management System itself.

* Time constraints requiring efficient test automation
* Testing environment limited to local development setup
* Testing database (hoa\_management\_test) separate from production
* Third party applications and packages are not subject to testing, only their integration and ability to achieve the desired results within the application.
* Actual payments are not handled by this application, so they will not be tested, only how they are handled internally.

## References:

* Project Requirements Document
* React Component Library
* API Service Documentation
* Database Schema

# Test Items:

The following sections identify the main test areas and associated components within the HOA Management System. Each component and feature outlined is essential to delivering secure, functional, and user-friendly experience for HOA members and administrators. Testing will ensure that these elements meet specified requirements, from frontend and backend functionalities to scheduled tasks and critical API endpoints, and that all integrations work seamlessly within the system.

## Features to be Tested:

### Frontend Components

#### The frontend of the web application, built with React JS, consists of public and private pages that handle user routing, display, and data interaction. Pages serve as main views for users, using components to collect, display, and update information in the database. Frontend testing will ensure responsiveness across screen sizes and consistent theming and styling.

#### Authentication Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Login.js | - Form validation - Email format validation - Password requirements Error handling Token storage - Session management |
| RegistrationModal.js | - Form validation - Data collection - Error handling - Success states- Account creation flow |
| ChangePasswordModal.js | - Password validation - Current password verification - Update process - Error states |
| ForgotPasswordModal.js | - Email validation - Reset token handling - Email trigger - Success/error states |
| ProtectedRoute.js | - Authentication checks - Role verification - Redirect handling |

#### Core Layout Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Layout.js | - Responsive rendering - Child component handling - Theme integration - Sidebar state |
| Header.js | - Navigation functionality - User menu - Theme toggle - Mobile responsiveness |
| Sidebar.js | - Navigation links - Collapse/expand - Active route highlighting - Role-based items - Mobile view |
| Footer.js | - Link functionality - Responsive design |

#### Dashboard Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Dashboard.js | - Data loading - Widget rendering - Error handling - Real-time updates |
| BoardMemberDashboard.js | - Permission checks - Action buttons - Data management - Role-specific features |

#### Owner Management Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| OwnerInfo.js | - Profile information display - Section management (Personal/Contact/Security) - Modal triggers - Error handling - Data refresh after updates |
| PersonalInfoModal.js | - Form validation - Data updates - Error handling - Success states - Field constraints |
| ContactInfoModal.js | - Contact information validation - Email format checking - Phone format validation - Update processing - Error states |
| NotificationPreferencesModal.js | - Preference toggles - Master email toggle - Individual notification settings - Save functionality - Current settings display - Enable/Disable all options - Error handling |
| BoardMemberSettingsModal.js | - Role management - Permission settings - Form validation - Current role display - Error handling - Success states |
| UpdateBoardMembersModal.js | - Member search - Role assignment - End date management - Validation r |

#### Communication Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Messages.js | - Message list - Conversation view - Search - Pagination |
| MessageDetail.js | - Content display - Reply handling - Attachments - User info |
| MessageList.js | - Preview display - Unread indicators - Sorting/filtering |
| NewMessageModal.js | - Recipient selection - Composition - Attachments - Sending |
| ReplyMessageModal.js | - Reply composition - Original quote - Sending |

#### Announcement Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| AnnouncementsNews.js | - List display - Creation interface - Filtering/search - Categories |
| AnnouncementModal.js | - Detail display - Media handling - Actions |
| CreateAnnouncementModal.js | - Form validation - Media upload - Preview - Publishing |
| AnnouncementCalendar.js | - Event display - Navigation - Event details |
| AnnouncementCarousel.js | - Slide functionality - Navigation - Responsiveness |

#### Financial Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Payments.js | - History display - Balance info - Filtering - Export |
| PaymentModal.js | - Amount input - Method selection - Processing - Receipts |
| CardManagementModal.js | - Card listing - Add/delete - Default setting |
| PaymentHistoryModal.js | - Transaction details - Receipt view - Printing |
| ChargeDetailsModal.js | - Charge info - Payment status - Documents |

#### Board Member Action Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| IssueViolationModal.js | - Type selection - Details input - Photo upload - Notice generation |
| IssueAssessmentModal.js | - Type selection - Amount calculation - Due dates - Resident selection |
| UpdateViolationsModal.js | - Type management - Fine schedule - Status updates |
| ManageAssessmentsModal.js | - Rate configuration - Schedule management - Bulk creation |

#### Document Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Documents.js | - Document list display - Category filtering - Search functionality - Sort operations - View type switching |
| DocumentCard.js | - Document info display - Action buttons - Download functionality - Status indicators |
| UploadDocumentModal.js | - File selection - Form validation - Upload progress - Error handling - Success states |
| ManageDocumentsModal.js | - Document list management - Edit capabilities - Delete operations - Access control settings |

#### Survey Components

|  |  |
| --- | --- |
| **Component** | **Test Requirements** |
| Survey.js | - Survey list display - Active/inactive filtering - Response collection - Results display |
| CreateSurveyModal.js | - Question creation - Answer options - End date setting - Validation rules |
| SurveyCompletionModal.js | - Answer selection - Submission handling - Validation - Success/error states |
| SurveyResultsModal.js | - Results visualization - Statistics display - Chart rendering - Data export |

### Backend Components

The backend is Node.js based and is comprised of a server, middleware, templates and routes for executing frontend calls and database changes.

#### Scheduled Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Schedule** | **Function** | **Test Requirements** |
| Survey Status Check | Daily at 00:00 | processSurveyResults | - Verify surveys past end date are processed - Check RESULTS\_SENT flag update - Validate transaction handling - Test error handling and rollback |
| Yearly Assessment | January 1st at 00:01 | yearlyAssessment | - Verify rate calculation - Check account balance updates - Test message creation - Validate email notifications - Test error handling |
| Announcement Check | Daily at 00:00 | announcementStatusCheck | - Verify scheduled announcements publish - Test status updates - Validate date/time handling |
| Voting Rights Check | Daily at 00:00 | votingRightsCheck | - Test suspension for past due accounts - Verify rights restoration - Validate message creation - Check notification delivery |
| Past Due Reminders | Weekly Monday at 09:00 | pastDueReminders | - Test reminder message creation - Verify email notifications - Validate 7-day reminder logic - Check balance calculations |

#### Complete API Routes and Functions Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| **Route** | **Function** | **Service Type** | **Test Requirements** |
| POST /login | login | authService | - Email/password validation - Token generation - Error handling |
| POST /refresh-token | refreshToken | authService | - Token validation - Token refresh - Error handling |
| POST /register | register | authService | - Data validation - Account creation - Error handling |
| POST /forgot-password | forgotPassword | authService | - Email validation - Token generation - Email sending |
| POST /reset-password | resetPassword | authService | - Token validation - Password update - Error handling |
| POST /verify-registration | verifyRegistration | authService | - Code validation - Account activation - Error handling |
| GET /profile | getProfile | apiService | - Authentication check - Data retrieval - Error handling |
| GET /dashboard | getDashboard | apiService | - Data aggregation - Permission check - Error handling |
| GET /account-details | getAccountDetails | apiService | - Account validation - Data retrieval - Error handling |
| GET /cards/:accountId | getCards | apiService | - Account verification - Card retrieval - Error handling |
| POST /cards/:accountId | addCard | apiService | - Card validation - Card creation - Error handling |
| PUT /cards/:cardId | updateCard | apiService | - Card validation - Update handling - Error handling |
| DELETE /cards/:cardId | deleteCard | apiService | - Card verification - Deletion handling - Error handling |
| PUT /cards/:cardId/default | setDefaultCard | apiService | - Card verification - Default setting - Error handling |
| POST /payments | submitPayment | apiService | - Payment validation - Transaction processing - Receipt generation |
| GET /payments/history | getPaymentHistory | apiService | - Authentication check - History retrieval - Error handling |
| GET /messages | getMessages | apiService | - Message retrieval - Pagination - Error handling |
| POST /messages | sendMessage | apiService | - Message validation - Message creation - Error handling |
| DELETE /messages/:messageId | deleteMessage | apiService | - Message verification - Deletion handling - Error handling |
| GET /users | searchUsers | apiService | - Search parameters - Results filtering - Error handling |
| GET /messages/thread/:messageId | getMessageThread | apiService | - Thread retrieval - Message ordering - Error handling |
| PUT /messages/:messageId/read | markMessageAsRead | apiService | - Message verification - Status update - Error handling |
| GET /announcements | getAnnouncements | apiService | - Announcement retrieval - Filtering - Error handling |
| POST /announcements | createAnnouncement | apiService | - Data validation - File handling - Error handling |
| PUT /announcements/:id | updateAnnouncement | apiService | - Update validation - File handling - Error handling |
| DELETE /announcements/:id | deleteAnnouncement | apiService | - Delete verification - File cleanup - Error handling |
| GET /documents | getDocuments | apiService | - Document retrieval - Access control - Error handling |
| POST /documents | uploadDocument | apiService | - File validation - Upload handling - Error handling |
| PUT /documents/:id | updateDocument | apiService | - Document validation - Update handling - Error handling |
| DELETE /documents/:id | deleteDocument | apiService | - Delete verification - File removal - Error handling |
| GET /documents/:id/download | downloadDocument | apiService | - Access verification - File streaming - Error handling |
| GET /surveys | getSurveys | apiService | - Survey retrieval - Status filtering - Error handling |
| POST /surveys | createSurvey | apiService | - Data validation - Survey creation - Error handling |
| POST /surveys/:id/responses | submitSurveyResponse | apiService | - Response validation - Answer recording - Error handling |
| GET /surveys/:id/results | getSurveyResults | apiService | - Results calculation - Data aggregation - Error handling |
| GET /violation-types | getViolationTypes | apiService | - Type retrieval - Permission check - Error handling |
| POST /violation-types | createViolationType | apiService | - Type validation - Creation handling - Error handling |
| PUT /violation-types/:id | updateViolationType | apiService | - Update validation - Permission check - Error handling |
| GET /active-owners | getActiveOwners | apiService | - Owner retrieval - Status filtering - Error handling |
| POST /violations | issueViolation | apiService | - Violation validation - Notice generation - Error handling |
| GET /assessments/types | getAssessmentTypes | apiService | - Type retrieval - Permission check - Error handling |
| GET /assessments/rates | getAssessmentRates | apiService | - Rate retrieval - Permission check - Error handling |
| POST /assessments/types/batch | addAssessmentTypes | apiService | - Batch validation - Creation handling - Error handling |
| POST /assessments/rates/batch | addAssessmentRates | apiService | - Rate validation - Creation handling - Error handling |
| POST /assessments/issue | issueAssessments | apiService | - Assessment validation - Charge creation - Error handling |
| GET /verify-board-member | verifyBoardMember | apiService | - Role verification - Permission check - Error handling |
| GET /board-members/roles | getBoardMemberRoles | apiService | - Role retrieval - Permission check - Error handling |
| PUT /board-members/roles/:id | updateBoardMemberRole | apiService | - Role validation - Update handling - Error handling |
| POST /board-members/roles | addBoardMemberRole | apiService | - Role validation - Creation handling - Error handling |
| GET /board-members/active | getActiveBoardMembers | apiService | - Member retrieval - Status check - Error handling |
| PUT /board-members/map/:ownerId/:memberId/end | endBoardMemberRole | apiService | - Role verification - End date setting - Error handling |
| POST /board-members | addBoardMember | apiService | - Member validation - Role assignment - Error handling |
| GET /properties/available | getAvailableProperties | apiService | - Property retrieval - Status filtering - Error handling |
| POST /accounts/create | createNewAccount | apiService | - Account validation - Creation handling - Error handling |
| GET /owner/details | getOwnerDetails | apiService | - Owner validation - Data retrieval - Error handling |
| PUT /owner/personal-info | updateOwnerPersonalInfo | apiService | - Data validation - Update handling - Error handling |
| PUT /owner/contact-info | updateOwnerContactInfo | apiService | - Contact validation - Update handling - Error handling |
| PUT /owner/notification-preferences | updateNotificationPreferences | apiService | - Preference validation - Update handling - Error handling |

## Features Not to Be Tested:

Content within public pages, contact details, and database-specific information will not be examined beyond consistency. The payment module will be tested only to ensure it completes the required database operations, as no external payments are processed. The email functionality, handled by SendGrid, will be verified solely by checking whether emails are sent and successfully received, without further testing of service functionality.

## Approach:

The overall approach to testing will be a mix of both manual testing and automated testing when it makes sense for it.

### Manual Testing

|  |  |
| --- | --- |
| **Category** | **Testing Requirements** |
| User Interface | * Cross-browser compatibility * Mobile responsiveness * Visual consistency * Navigation flow |
| Integration Points | * Email receipt delivery * PDF generation * File upload/download * Payment confirmation |
| Security | * Access control verification * Session handling * Input validation |

### Automated Testing

|  |  |  |
| --- | --- | --- |
| **Category** | **Tools** | **Testing Requirements** |
| Unit Tests | Jest / React Testing Library | Testing individual components for input/output |
| Integration Tests | Cypress | Testing input/output of a flow or sequence |
| API Test | Jest / Supertest / Postman | API endpoint functionality |

## Item Pass/Fail Criteria:

### Pass Criteria:

* All automated and manual tests pass, including routes, visuals, and responsiveness.
* No critical or high-priority bugs are present.
* Core functionality operates as specified.

### Fail Criteria:

* Critical functionality or features are missing or fail.
* Security vulnerabilities are identified.
* Data integrity issues are present.

## Suspension Criteria and Resumption Requirements:

Testing will be suspended under the following conditions:

* Critical Functionality Missing: If any core functionalities essential to the HOA Management System’s operation are found to be incomplete or non-operational, testing will pause until those features are implemented.
* Database Integrity Issues: If testing reveals corruption or integrity issues within the test database, all tests that could impact data integrity will be halted. Testing will resume once the database integrity is fully restored, and safeguards are in place to prevent further issues.

Testing will only continue once all critical functionality has been implemented, and data integrity is verified in the test environment.

## Test Deliverables:

* Test Plan (this document).
* Requirements Traceability Matrix
* Test cases for functional and non-functional features.
* Test scripts (for automated testing).
* Test logs documenting defects and their resolution.
* Test reports summarizing pass/fail rates, issues, and recommendations.

## Test Environment:

* **Frontend:** React (latest stable version).
* **Backend:** Node.js, MySQL database.
* **Testing Tools:** Jest, React Testing Library, Cypress, Postman (for API testing), Supertest, and Chrome DevTools
* **Environment:** Tested on Chrome, Firefox, and mobile devices for compatibility.

### Hardware Requirement:

* Development machine with minimum 8GB RAM
* Stable internet connection
* Multiple test devices for responsive testing (Browser tools)

### Software Requirements

* Node.js latest LTS version
* MySQL 8.0
* Modern browsers (Chrome, Firefox, Safari)
* VS Code with testing extensions

## Estimate:

The total time estimate for testing is approximately 8 days.

* Unit Testing: 2 days
* Integration Testing: 2 days
* System Testing: 2 days
* Acceptance Testing: 2 days

## Schedule:

* **Start Date:** November 4, 2024
* **End Date:** November 12, 2024
* Key Milestones:
  + Completion of Unit Testing: November 6, 2024
  + System Testing Completion: November 10, 2023
  + Final Acceptance Testing: November 12, 2023

## Staffing and Training Needs:

The staffing and training needs for testing consists of 1 developer.

## Responsibilities:

* The single developer will be responsible for all tasks and adherence to the testing plan

## Risks:

* Delayed API development: Could hinder system integration testing.
* Unforeseen bugs: Could extend the testing phase.
* Limited testing on real devices: Could limit identifying mobile responsiveness issues. Responsiveness will be tested on a variety of devices utilizing the capabilities inherent with browsers for display on a range of smaller devices

## Assumptions and Dependencies:

* The backend and frontend teams will deliver stable versions for testing on time.
* Test environment mirrors the production environment.
* External dependencies are functioning correctly.

## Approvals:

* Senior Project Advisor: Michael O’Neill
* Project Manager/Developer/Tester: Justin Pike